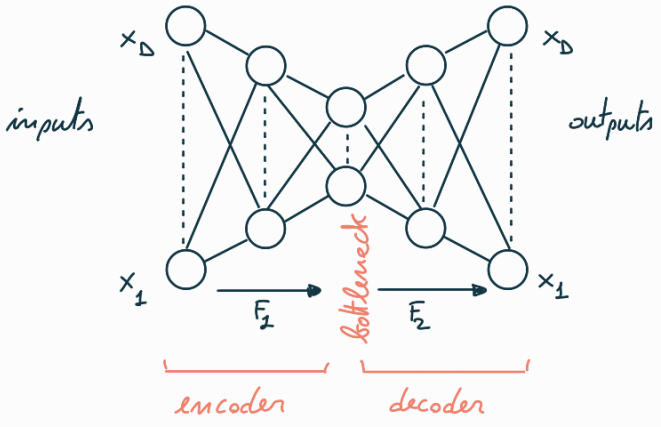
What happens when the latent space is not linear?

PCA makes linear transformations. Linear representations are not sufficient for complex data.
Non-linear models seen so far: neural networks.
Using a particular layout of neural networks we can solve dimensionality reduction problems in which the transformation from the original to the latent space is non-linear. To do this we use autoencoders.

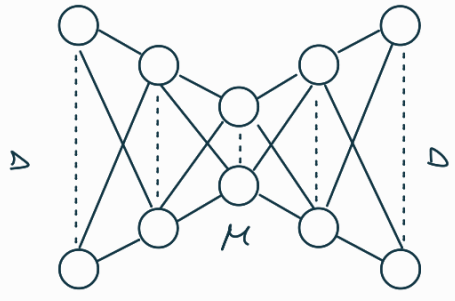
What is an autoencoder?

An autoencoder is a combination of two Neural Networks: encoder + decoder. They are structured this way in order to introduce a bottleneck.



dimension of the input = dimension of the output = dimension of the dataset

The bottleneck will reduce the dimensionality



We can have as many layers as we want between the input/output and the middle ⟶ deep autoencoders
The layers will contain non-linear activation functions: the transformation from D to M and then from M to D is non-linear.

An ideal model will perfectly reconstruct the input vector ⟶ $x_m = x'_m$

z_m is an intermediate representation of x_m of M dimensions.

Obviously we want to minimize the difference between x_n and x'_n. If we find the parameters of the network such that this is guaranteed, we learned how to represent the dataset in a lower dimensional space. This process is called reconstruction loss (using the input as ground truth after a reconstruction) → autoencoders are trained using the same sample both in input and as ground truth.

Once we trained the network, we can:

- use only the encoder to encode new data
- use only the decoder to generate new data → generate some value in the latent space and construct new data.

We can apply this concept to any kind of data

e.g. convolutional autoencoders = autoencoders containing convolutional layers
Autoencoders for images are based on Convolutional and Convolutional Transposed layers.

Autoencoders are not for classification problems, so to train them we use either the mean square error or the mean absolute error, metrics based on the reconstruction error.

Autoencoders for anomaly detection (one-class classification)

Autoencoders are very useful for anomaly detection (one-class classification).
Anomaly detection is a situation in which we have to distinguish between normal and abnormal.

Problem:
learn $f: X \to \{n, a\}$ with dataset $\Delta = \{(x_n, n)\}$
(n: normal, a: abnormal; we assume to have only normal data in the dataset)
At training time we only have data for one class but we still want to learn a function that after training can distinguish between the two. This problem can be extended to multiple classes.

Solution: (consider the data as unsupervised)
- train an autoencoder with $\{x_n\}$ (learn a latent representation for normal samples)
- after training we reach a loss
- determine a threshold δ, e.g. $\delta = \text{mean}(\text{loss}) + \text{std}(\text{loss})$, $\delta = \max(\text{loss})$, ...
- given $x' \notin \Delta$, reconstruct x' with AE and compute loss'.
- if $\text{loss}' < \delta$ (loss' is within the threshold) return normal, otherwise return abnormal.

If x' has not been well reconstructed by the autoencoder, it means that it is different in some sense with respect to the other samples in Δ. We know this since the loss is high, thus the AE was not good on the sample and this can be considered as an anomaly.

This works even better if instead of using a single autoencoder we use an ensemble (iterate the procedure with different autoencoders, possibly even with different structure)